Optimizing thermodynamic trajectories using evolutionary reinforcement learning

Chris Beeler[1],* Uladzimir Yahorau[1], Rory Coles[1], Kyle Mills[1], Stephen Whitelam[2], and Isaac Tamblyn[1,3,4]†
[1] *University of Ontario Institute of Technology, Oshawa, ON, Canada*
[2] *Molecular Foundry, Lawrence Berkeley National Laboratory, Berkeley, CA, USA*
[3] *University of Ottawa, Ottawa, ON, Canada*
[4] *National Research Council of Canada, Ottawa, ON, Canada*
(Dated: March 21, 2019)

Using a model heat engine we show that neural network-based reinforcement learning can identify thermodynamic trajectories of maximal efficiency. We use an evolutionary learning algorithm to evolve a population of neural networks, subject to a directive to maximize the efficiency of a trajectory composed of a set of elementary thermodynamic processes; the resulting networks learn to carry out the maximally-efficient Carnot, Stirling, or Otto cycles. Given additional irreversible processes this evolutionary scheme learns a hitherto unknown thermodynamic cycle. Our results show how the reinforcement learning strategies developed for game playing can be applied to solve physical problems conditioned upon path-extensive order parameters.

Introduction – Games, whether played on a board, such as chess or Go, or played on the computer, are a major component of human culture [1]. In the language of physics, games are *trajectories*, time-ordered sequences of elementary steps. The outcome of a game is a path-extensive order parameter, one determined by the entire history of the trajectory. Playing games was once the preserve of human beings, but machine-learning methods now outperform the most talented humans in all the aforementioned examples [2–19]. Motivated by the correspondence between games and trajectories, it is natural to ask how the machine-learning methods that have mastered game-playing might be applied to understand physical processes whose outcomes are path-extensive quantities.

There are many examples of such processes. For instance, the success or failure of molecular self-assembly is determined by a time history of elementary dynamical processes, including the binding and unbinding of particles [20–23]. Dynamical systems, such as chemical networks and molecular machines [24–27], are characterized by time-extensive observables, such as work or entropy production [28–34]. In none of these cases do we possess a complete theoretical or practical understanding of how to build an arbitrary structure or maximize the efficiency of an arbitrary machine. Traditional methods of inquiry in physics focus on applying physical intuition and the manipulation and simulation of equations; perhaps machine learning can provide us with further insight into physical problems of a path-extensive nature.

Motivated by this speculation, we show here that neural network-based reinforcement learning can maximize the efficiency of the simplest type of physical trajectories, the deterministic, quasi-static ones of classical thermodynamics. We introduce a model heat engine characterized by a set of thermodynamic state variables. A neural network takes as input the current microstate of the engine and chooses one of a set of basic thermodynamic processes to produce a new microstate; this change comprises one step of a trajectory. We generate a set of trajectories of fixed length using a set of networks whose parameters are initially randomly chosen, and retain and mutate only those networks whose trajectories show the greatest efficiency. Repeating this evolutionary process many times results in networks whose trajectories reproduce the maximally efficient Carnot, Stirling, or Otto cycles, depending upon which basic thermodynamic processes are allowed. This evolutionary procedure can also learn previously unknown thermodynamic cycles if new processes are allowed. The present approach shows how to adapt the machine-learning techniques developed for game-playing to thermodynamic trajectories, and points the way to the generalization of this approach to a wide variety of physical trajectories.

Model heat engine and thermodynamic trajectories – In Fig. 1(a) we show a model heat engine, a device able to transform thermal energy into work [35, 36]. The engine consists of a working substance, which we assume to be a monatomic ideal gas, housed within a container of variable volume V, whose minimum and maximum values are $V_{\rm min}$ and $V_{\rm max}$, respectively. The working substance may be connected to a hot or cold reservoir held at temperature $T_{\rm h} = 500$ K and $T_{\rm c} = 300$ K, respectively, or may be insulated. The instantaneous microstate x of the system is then specified by the volume-temperature vector $x = (V, T)$, with the pressure of the system fixed by the ideal-gas equation $PV = Nk_{\rm B}T$ [37].

To evolve the heat engine we use the neural network shown in Fig. 1(b). The network is a nonlinear function that takes as input the current microstate x of the system, and outputs the probabilities $p_y(x;\theta)$ of moving to any of a set of M new microstates $y \in \{y_1, \ldots, y_M\}$ (in the language of reinforcement learning this mapping is called a *policy* [38]). The symbol θ denotes the internal parameters of the network, discussed shortly. In this paper we consider deterministic evolution through con-

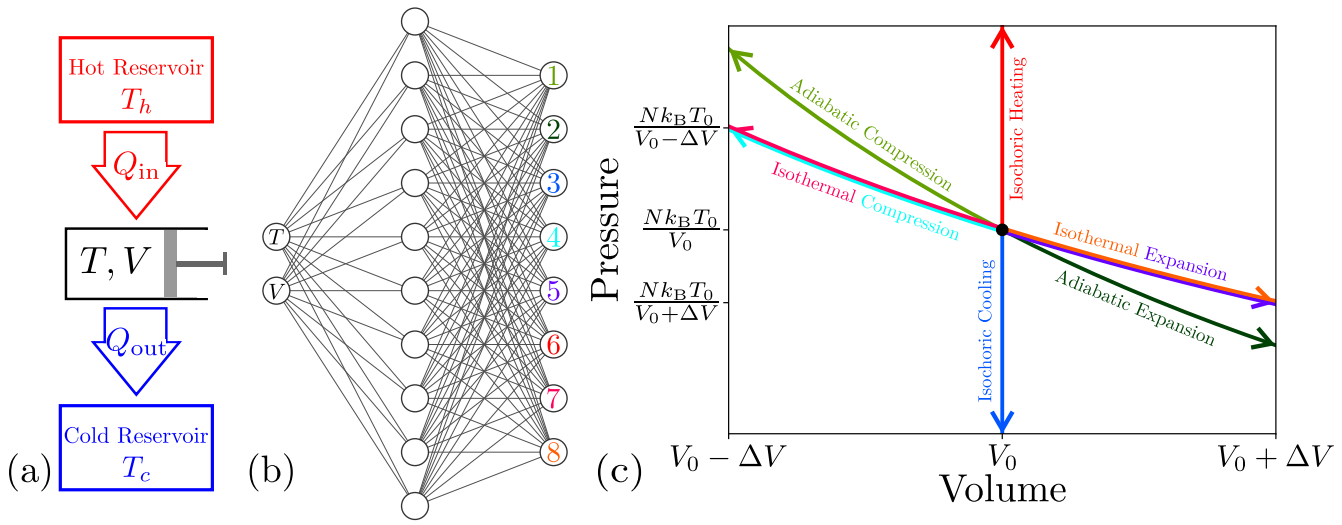

FIG. 1. (a) Model heat engine and (b) the neural network that evolves it. (c) A summary of the actions in P-V space available to the network; see Table 1.

figuration space, with $p_{y^\star}(x;\theta)$ equal to 1 for a chosen process $x \to y^\star$, and equal to zero otherwise. Enacting the chosen process corresponds to one step of a trajectory. Given an initial microstate x_0, K applications of the network produces a trajectory $\omega = x_0 \to x_1 \to \cdots \to x_K$ of K steps through configuration space. In this paper we focus on trajectories of $K = 200$ steps.

The elementary moves available to the network correspond to the basic thermodynamic processes shown in Table 1, summarized graphically in Fig. 1(c). These processes include compression and expansion, along isotherms or adiabats, and temperature changes along isochores. Upon making any move $x \to y$ we record the resulting changes of work, ΔW_{xy}, and heat input from the hot reservoir, $\Delta Q_{xy} H(\Delta Q_{xy}\delta_{T_{\rm f},T_{\rm h}})$; these are listed in Table 1. Here $H(\cdot)$ is the Heaviside function, equal to 1 for positive values of ΔQ_{xy} and 0 otherwise, and $T_{\rm f}$ is the temperature of the system following the move. $\delta_{\alpha,\beta}$ is the Kronecker delta symbol, equal to 1 if $\alpha = \beta$ and 0 otherwise. We define the thermodynamic efficiency of a K-step trajectory as

$$\eta_K \equiv \frac{\sum_{k=0}^{K-1} \Delta W_{x_k x_{k+1}}}{\sum_{k=0}^{K-1} \Delta Q_{x_k x_{k+1}} H(\Delta Q_{x_k x_{k+1}} \delta_{T_{\rm f},T_{\rm h}})}. \quad (1)$$

The efficiency, a path-extensive quantity, is used as a means of choosing between trajectories, and the networks that generate them, during our evolutionary learning procedure. Given that trajectories are deterministic, the maximum value $\eta = \max_K \eta_K$ at any point along a long trajectory (excluding values at early times) is sufficient to identify efficient thermodynamic cycles.

The network, which contains two layers of tunable weights, performs computations as follows. Two input neurons receive the current microstate x, and the output is comprised of $M \leq 8$ neurons, each corresponding to one of the actions shown in Table 1 (in some simulations we prohibit certain actions). The network possesses one hidden layer of 1024 neurons, each connected to every input and output neuron. Let the indices $i \in \{1,2\}$, $j \in \{1,\ldots,1024\}$, and $k \in \{1,\ldots,M\}$ label the neurons of the input, hidden, and output layers, respectively. The input I_i of the two nodes $i = 0,1$ of the input layer are, respectively, scaled versions of the current temperature $(T - T_{\rm c}) / (T_{\rm h} - T_{\rm c}) \in [0,1]$ and volume $(V - V_{\rm min}) / (V_{\rm max} - V_{\rm min}) \in [0,1]$ of the system. We set the output signal S_i of each input-layer node as $S_i = I_i$.

The input I_j to neuron j in the hidden layer is

$$I_j = \sum_{i=1}^{2} S_i w_{ij}, \quad (2)$$

where the sum runs over the two neurons in the input layer, and w_{ij} is the weight of the connection between nodes i and j. We set the output signal S_j of neuron j to be

$$S_j = \frac{1}{2}\left[\tanh\left(I_j + b_j\right)\right], \quad (3)$$

where b_j is a bias associated with neuron j.

The input I_k to neuron k in the output layer is

$$I_k = \sum_{j=1}^{1024} S_j w_{jk}, \quad (4)$$

where the sum runs over all 1024 neurons of the hidden layer. Finally, we take the output signal S_k from each output-layer neuron to be equal to I_k. To choose an action we pick the output neuron, $k^\star$, with the largest

Action	ΔW	ΔQ
Adiabatic Compression	$-\frac{3}{2}Nk_{\mathrm{B}}T_{\mathrm{i}}\left(\left(\frac{V_{\mathrm{i}}}{V_{\mathrm{f}}}\right)^{\frac{2}{3}}-1\right)$	0
Adiabatic Expansion	$-\frac{3}{2}Nk_{\mathrm{B}}T_{\mathrm{i}}\left(\left(\frac{V_{\mathrm{i}}}{V_{\mathrm{f}}}\right)^{\frac{2}{3}}-1\right)$	0
Isothermal Compression at T_{h} ($T=T_{\mathrm{h}}$)	$Nk_{\mathrm{B}}T_{\mathrm{h}}\log\left(\frac{V_{\mathrm{f}}}{V_{\mathrm{i}}}\right)$	$Nk_{\mathrm{B}}T_{\mathrm{h}}\log\left(\frac{V_{\mathrm{f}}}{V_{\mathrm{i}}}\right)$
Isothermal Expansion at T_{h} ($T=T_{\mathrm{h}}$)	$Nk_{\mathrm{B}}T_{\mathrm{h}}\log\left(\frac{V_{\mathrm{f}}}{V_{\mathrm{i}}}\right)$	$Nk_{\mathrm{B}}T_{\mathrm{h}}\log\left(\frac{V_{\mathrm{f}}}{V_{\mathrm{i}}}\right)$
Isothermal Compression at T_{h} ($T\neq T_{\mathrm{h}}$)	$Nk_{\mathrm{B}}T_{\mathrm{h}}\log\left(\frac{V_{\mathrm{f}}}{V_{\mathrm{i}}}\right)$	$Nk_{\mathrm{B}}T_{\mathrm{h}}\log\left(\frac{V_{\mathrm{f}}}{V_{\mathrm{i}}}\right)+\frac{3}{2}Nk_{\mathrm{B}}\left(T_{\mathrm{h}}-T_{\mathrm{i}}\right)$
Isothermal Expansion at T_{h} ($T\neq T_{\mathrm{h}}$)	$Nk_{\mathrm{B}}T_{\mathrm{h}}\log\left(\frac{V_{\mathrm{f}}}{V_{\mathrm{i}}}\right)$	$Nk_{\mathrm{B}}T_{\mathrm{h}}\log\left(\frac{V_{\mathrm{f}}}{V_{\mathrm{i}}}\right)+\frac{3}{2}Nk_{\mathrm{B}}\left(T_{\mathrm{h}}-T_{\mathrm{i}}\right)$
Isothermal Compression at T_{c} ($T=T_{\mathrm{c}}$)	$Nk_{\mathrm{B}}T_{\mathrm{c}}\log\left(\frac{V_{\mathrm{f}}}{V_{\mathrm{i}}}\right)$	$Nk_{\mathrm{B}}T_{\mathrm{c}}\log\left(\frac{V_{\mathrm{f}}}{V_{\mathrm{i}}}\right)$
Isothermal Expansion at T_{c} ($T=T_{\mathrm{c}}$)	$Nk_{\mathrm{B}}T_{\mathrm{c}}\log\left(\frac{V_{\mathrm{f}}}{V_{\mathrm{i}}}\right)$	$Nk_{\mathrm{B}}T_{\mathrm{c}}\log\left(\frac{V_{\mathrm{f}}}{V_{\mathrm{i}}}\right)$
Isothermal Compression at T_{c} ($T\neq T_{\mathrm{c}}$)	$Nk_{\mathrm{B}}T_{\mathrm{c}}\log\left(\frac{V_{\mathrm{f}}}{V_{\mathrm{i}}}\right)$	$Nk_{\mathrm{B}}T_{\mathrm{c}}\log\left(\frac{V_{\mathrm{f}}}{V_{\mathrm{i}}}\right)$
Isothermal Expansion at T_{c} ($T\neq T_{\mathrm{c}}$)	$Nk_{\mathrm{B}}T_{\mathrm{c}}\log\left(\frac{V_{\mathrm{f}}}{V_{\mathrm{i}}}\right)$	$Nk_{\mathrm{B}}T_{\mathrm{c}}\log\left(\frac{V_{\mathrm{f}}}{V_{\mathrm{i}}}\right)$
Isochoric Heating	0	$\frac{3}{2}Nk_{\mathrm{B}}\left(T_{\mathrm{h}}-T_{\mathrm{i}}\right)$
Isochoric Cooling	0	$\frac{3}{2}Nk_{\mathrm{B}}\left(T_{\mathrm{c}}-T_{\mathrm{i}}\right)$

TABLE I. All possible actions that can be taken on our model heat engine and their corresponding ΔW and ΔQ equations.

value of S_k. Given a current microstate x, this action defines a new microstate $y^\star$ via Table 1. The probability $p_{y^\star}(x;\theta)$ is then unity, and all other $p_y(x;\theta)$ are zero. We denote by $\theta=\{\{w\},\{b\}\}$ the set of all weights and biases of the network. Initially each weight and bias is chosen from a Gaussian distribution with zero mean and unit variance.

Evolutionary learning dynamics – With the thermodynamic system and means of evolving it defined, we introduce an evolutionary learning dynamics designed to produce networks able to propagate efficient thermodynamic trajectories. We start with a population of 100 networks, with the internal parameters θ of each initialized in the random fashion described above. We name this population generation 1. This population produces thermodynamic trajectories ω of K steps with the distribution $P(\eta)$ of efficiencies η shown in Fig. 2(a). Even the best-performing members of this population produce efficiencies much lower than the Carnot efficiency $\eta_C=1-T_{\mathrm{c}}/T_{\mathrm{h}}=2/5$, which is the most efficient trajectory possible given the set of allowed thermodynamic processes [36].

We next perform the first step of evolutionary learning dynamics. We keep the 25 generation-1 networks whose trajectories have the largest η, and we discard the rest. We create 75 new networks by drawing uniformly from the set of 25, each time "mutating" all weights w and biases b: for each weight or bias we draw a random number δ from a Gaussian distribution with zero mean and unit variance, and update the weight or bias as $w\to w+\epsilon\delta$ or $b\to b+\epsilon\delta$, where $\epsilon=0.05$ is an evolutionary learning rate.

The new population of the 25 best generation-1 networks and their 75 mutant offspring constitute generation 2. We simulate those 100 networks for K steps, producing the distribution of efficiencies shown in Fig. 2(a). Continuing this alternation of evolutionary dynamics (retaining and mutating the best networks of the current generation) and physical dynamics (using the new generation of networks to generate a set of trajectories) gives rise to networks able to propagate increasingly efficient trajectories [Fig. 2(a)]. After about 100 generations, we obtain networks whose efficiencies are equal to that of the Carnot cycle to within four decimal places. Inspection of the trajectories corresponding to these values of η show that they indeed form Carnot cycles; see Fig. 2(b).

Several features of this learning process are notable. In learning to maximize the efficiency of a thermodynamic trajectory, networks have learned to exclude the isochoric processes listed in Table 1, which do not appear in the Carnot cycle. Networks have also learned to propagate cycles, as opposed to non-closed loops in P-V space, because cycles lead in general to larger efficiencies. The Carnot cycle has no absolute scale associated with it; given the discrete step sizes permitted (Table 1), networks have learned to enact the size of a cycle that best approximates a closed loop (because closed loops have greatest efficiency).

Given only a set of processes and a path-extensive measure of efficiency, our neural network-based evolutionary learning framework is able to maximize path efficiency and so deduce a classic result of physics. This learning framework is similarly successful if it is presented with a different set of processes. When denied the adiabatic processes of Table 1 it learns the Stirling cycle [39], which is maximally efficient in this context; when denied the isothermal processes it learns the maximally-efficient Otto cycle [40]; see Fig. S3.

Extensions to unknown thermodynamic processes are straightforward, and inspection of the resulting solutions

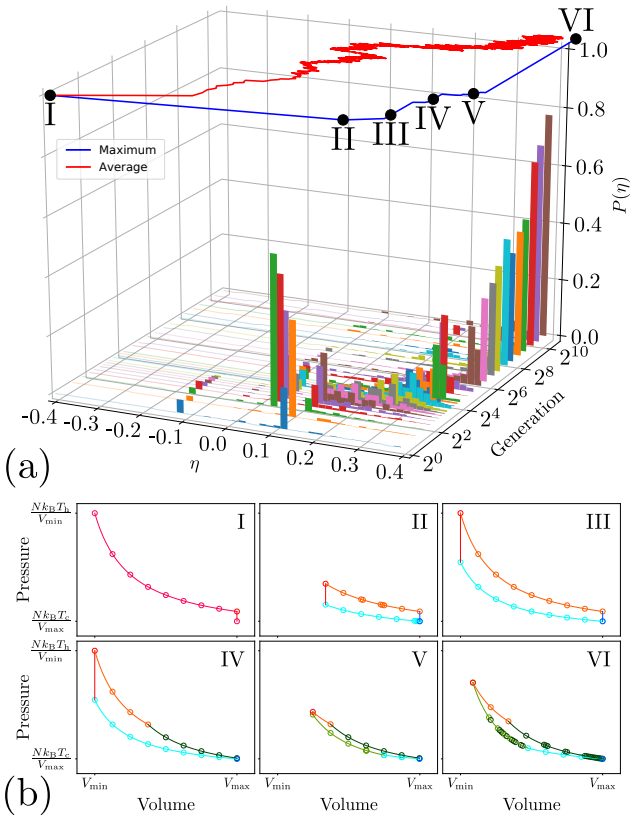

FIG. 2. (a) The evolution, as a function of generation number, of the probability distribution $P(\eta)$ of efficiencies η of trajectories of the model heat engine. The maximum and average efficiency of the population are shown above. The Carnot efficiency is $\eta_C = 0.4$. (b) Trajectories in P-V space produced by the best-performing networks in generations 2^0, 2^1, 2^2, 2^4, 2^5, and 2^{12}, in the boxes labeled I-VI, respectively. The colors of the branches correspond to the processes shown in Fig. 1. Highly-evolved networks enact the Carnot cycle.

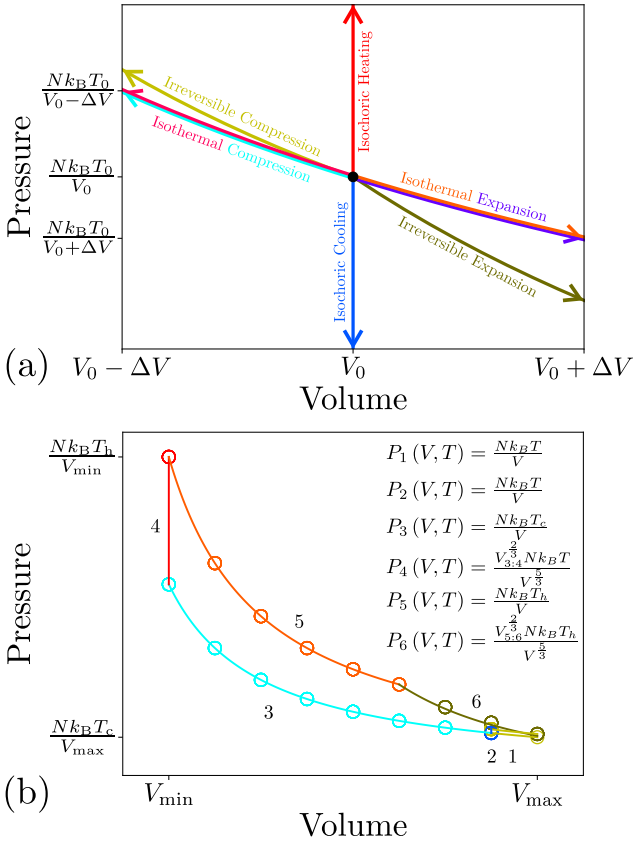

FIG. 3. We apply the evolutionary process described in Fig. 2 to a new setting in which the adiabatic processes of Table 1 are replaced with the fictitious process (5); panel (a) summarizes the new set of accessible moves. (b) Highly-evolved networks learn to enact a hybrid of the Stirling and Carnot cycles, and the resulting equations of state can be identified by curve fitting.

provides physical insight in an unfamiliar setting. As an illustration, we replace the standard monatomic ideal gas adiabatic process, for which $TV^{2/3} = \text{const.}$, with a fictitious irreversible process for which

$$TV^{2/3} \propto (1-k)^{\Delta V/(V_0-V_1)}; \qquad (5)$$

here $k = 2/5$ and ΔV are the fraction of thermal energy lost and the volume change upon making the move, respectively. We allow the network access to this process and the others of Table 1 (excluding the adiabatic processes), summarized in Fig. 3(a). In this setting we do not know in advance the most efficient trajectory. In Fig. 3(b) we show that the solution identified by our evolutionary learning scheme is a hybrid of the Stirling and Carnot cycles. By fitting equations to each branch of the cycle we identify the equations of state that result from the fictitious process (5). These results highlight the general applicability of the learning scheme and indicate the physical insight that can be obtained by interrogating solutions identified by machine learning.

Conclusions – Motivated by the correspondence between games and physical trajectories, we have shown that neural network-based evolutionary learning can optimize the efficiency of trajectories of classical thermodynamics. Given a set of physical processes and a path-extensive measure of efficiency, networks evolve to learn the maximally-efficient Carnot, Stirling, or Otto cycles, reproducing classic results of physics that were originally derived by application of physical insight. Given new processes, the evolutionary framework identifies solutions that when interrogated provide physical insight into the problem at hand.

Our results point the way to the application of evolutionary learning to a wide variety of physical trajectories. For instance, the scheme shown in Fig. 1(b) generalizes naturally to Markov Chain Monte Carlo simulation [41], with the neurons of the output layer corresponding to

members of a set of possible processes, and the normalized output $S_k/\sum_k S_k$ corresponding to the probability of choosing that process. Moreover, the scheme is numerically robust, requiring only a path-extensive order parameter to act as an evolutionary "pressure" via a series of discrete decisions. It does not require the smoothness of this order parameter as a function of network parameters or trajectory dynamics, as would be the case for gradient-based reinforcement learning methods that use backpropagation [3, 6]. The present approach can therefore be applied to physical problems in which the quality of the trajectory varies in a sudden or abrupt way. Such is the case in self-assembly, for instance, where the inclusion or omission of a single microscopic move may cause the yield of a target structure to jump abruptly.

CB, UY, RC, KM, and IT performed work at the National Research Council of Canada. SW performed work at the Molecular Foundry, Lawrence Berkeley National Laboratory, supported by the Office of Science, Office of Basic Energy Sciences, of the U.S. Department of Energy under Contract No. DE-AC02–05CH11231.

The heat engine environments [15] used in this study can be found at:

`https://github.com/CLEANit/heatenginegym`

* christopher.beeler@uoit.net
† isaac.tamblyn@nrc.ca

[1] J. M. Roberts, M. J. Arth, and R. R. Bush, American anthropologist **61**, 597 (1959).
[2] C. J. Watkins and P. Dayan, Machine learning **8**, 279 (1992).
[3] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, in *NIPS Deep Learning Workshop* (2013).
[4] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, Nature **518**, 529 (2015).
[5] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, Journal of Artificial Intelligence Research **47**, 253 (2013).
[6] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, in *International conference on machine learning* (2016) pp. 1928–1937.
[7] Y. Tassa, Y. Doron, A. Muldal, T. Erez, Y. Li, D. d. L. Casas, D. Budden, A. Abdolmaleki, J. Merel, A. Lefrancq, *et al.*, arXiv preprint arXiv:1801.00690 (2018).
[8] E. Todorov, T. Erez, and Y. Tassa, in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on* (IEEE, 2012) pp. 5026–5033.
[9] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming* (John Wiley & Sons, 2014).
[10] A. Asperti, D. Cortesi, and F. Sovrano, "Crawling in rogue's dungeons with (partitioned) a3c: 4th international conference, lod 2018, volterra, italy, september 13-16, 2018, revised selected papers," (2019) pp. 264–275.
[11] M. Riedmiller, in *European Conference on Machine Learning* (Springer, 2005) pp. 317–328.
[12] M. Riedmiller, T. Gabel, R. Hafner, and S. Lange, Autonomous Robots **27**, 55 (2009).
[13] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, arXiv preprint arXiv:1707.06347 (2017).
[14] F. P. Such, V. Madhavan, E. Conti, J. Lehman, K. O. Stanley, and J. Clune, in *NIPS Deep Reinforcment Learning Workshop* (2018).
[15] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, arXiv preprint arXiv:1606.01540 (2016).
[16] M. Kempka, M. Wydmuch, G. Runc, J. Toczek, and W. Jaśkowski, in *Computational Intelligence and Games (CIG), 2016 IEEE Conference on* (IEEE, 2016) pp. 1–8.
[17] M. Wydmuch, M. Kempka, and W. Jaśkowski, IEEE Transactions on Games (2018).
[18] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et al.*, nature **529**, 484 (2016).
[19] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, *et al.*, Nature **550**, 354 (2017).
[20] J. J. De Yoreo, P. U. Gilbert, N. A. Sommerdijk, R. L. Penn, S. Whitelam, D. Joester, H. Zhang, J. D. Rimer, A. Navrotsky, J. F. Banfield, *et al.*, Science **349**, aaa6760 (2015).
[21] M. F. Hagan and D. Chandler, Biophysical Journal **91**, 42 (2006).
[22] A. W. Wilber, J. P. Doye, A. A. Louis, E. G. Noya, M. A. Miller, and P. Wong, The Journal of Chemical Physics **127**, 085106 (2007).
[23] S. Whitelam and R. L. Jack, Annual review of physical chemistry **66**, 143 (2015).
[24] D. T. Gillespie, Annu. Rev. Phys. Chem. **58**, 35 (2007).
[25] T. McGrath, N. S. Jones, P. R. ten Wolde, and T. E. Ouldridge, Physical Review Letters **118**, 028101 (2017).
[26] U. Seifert, Reports on progress in Physics **75**, 126001 (2012).
[27] A. I. Brown and D. A. Sivak, Proceedings of the National Academy of Sciences **114**, 11057 (2017).
[28] U. Seifert, Physical Review Letters **95**, 040602 (2005).
[29] V. Lecomte, C. Appert-Rolland, and F. Van Wijland, Journal of statistical physics **127**, 51 (2007).
[30] F. Ritort, Advances in Chemical Physics **137**, 31 (2008).
[31] J. P. Garrahan, R. L. Jack, V. Lecomte, E. Pitard, K. van Duijvendijk, and F. van Wijland, Journal of Physics A: Mathematical and Theoretical **42**, 075007 (2009).
[32] T. Speck, A. Engel, and U. Seifert, Journal of Statistical Mechanics: Theory and Experiment **2012**, P12001 (2012).
[33] V. Lecomte, A. Imparato, and F. v. Wijland, Progress of Theoretical Physics Supplement **184**, 276 (2010).
[34] R. J. Harris, Journal of Statistical Mechanics: Theory and Experiment **2015**, P07021 (2015).
[35] H. B. Callen, *Thermodynamics and an introduction to thermostatistics*, 2nd ed. (John Wiley & Sons, New York, 1985).
[36] S. Carnot, Reflections on the motive power of fire by Sadi Carnot and other papers on the Second Law of Thermodynamics by E. Clapeyron and R (1824).

[37] M. S. Silberberg, *Principles of general chemistry* (McGraw-Hill Higher Education New York, 2007).
[38] R. S. Sutton, A. G. Barto, F. Bach, *et al.*, "Reinforcement learning: An introduction," (1998).
[39] T. Finkelstein, *Insights into the thermodynamic analysis of Stirling cycle machines*, Tech. Rep. (AIAA-94-3951-CP, 1829).
[40] M. Mozurkewich and R. S. Berry, Journal of Applied Physics **53**, 34 (1982).
[41] W. K. Hastings, (1970).

Supplementary Information for "Optimizing thermodynamic trajectories using evolutionary reinforcement learning"

Chris Beeler[1],* Uladzimir Yahorau[1], Rory Coles[1], Kyle Mills[1], Stephen Whitelam[2], and Isaac Tamblyn[1,3,4†]
[1] *University of Ontario Institute of Technology, Oshawa, ON, Canada*
[2] *Molecular Foundry, Lawrence Berkeley National Laboratory, Berkeley, CA, USA*
[3] *University of Ottawa, Ottawa, ON, Canada*
[4] *National Research Council of Canada, Ottawa, ON, Canada*
(Dated: March 21, 2019)

Background – In 1824, Carnot's theorem was developed, which states that the maximum thermal efficiency, $\eta_{\max}$, of any heat engine is dependent on the temperatures of the reservoirs and derived to be

$$\eta_{\max} = \frac{T_h - T_c}{T_h}. \tag{1}$$

$\eta_{\max}$ can only be achieved by performing a specific set of actions on the heat engine, which creates a cycle known as the Carnot cycle. This cycle is shown in Fig.1(a), with η for several cycles shown in Fig.2(a). Starting at the maximum volume, $V_{\max}$, the heat engine is compressed isothermally while connected to the cold reservoir until the engine approaches its minimum volume $V_{\min}$. Next the engine is adiabatically compressed until the temperature reaches T_h. During the compression steps the heat engine is performing work on the working substance, therefore η decreases during this part of the cycle. The engine, at $V_{\min}$, is then expanded isothermally while connected to the hot reservoir until the engine approaches $V_{\max}$. Finally the engine is adiabatically expanded until the temperature reaches T_c and $V_{\min}$, ending at the starting point of the cycle, extracting the most possible W given a fixed Q_h. During the expansion steps the working substance is performing work on the heat engine, therefore η increases during this part of the cycle, explaining the oscillating behavior seen in Fig.2.

The Stirling cycle is similar to the Carnot cycle; the major difference comes from replacing the adiabatic processes with isochoric processes. This cycle is shown in Fig.1(b), with η for several cycles shown in Fig.2(b). Staring at $V_{\max}$, the engine is compressed isothermally while connected to the cold reservoir until it reaches $V_{\min}$. Next the engine is connected to the hot reservoir, allowing the body to warm up isochorically to T_h. The engine is then expanded isothermally until it reaches $V_{\max}$. Finally the engine is connected to the cold reservoir, allowing the body to cool down isochorically to T_c. If a regenerative device is used to exchange internal heat which would otherwise be lost during the isochoric cooling, the Stirling thermal efficiency, η_{S}, is the same as $\eta_{\max}$. However, without such device, η_{S} is derived to be

$$\eta_{\mathrm{S}} = \frac{T_h - T_c}{T_h + \frac{\Delta U_V}{\Delta S_T}} \tag{2}$$

where ΔU_V is the change in internal energy for an isochoric process, and ΔS_T is the change in entropy for an isothermal process, defined respectively as

$$\Delta U_V = C_V \left(T_h - T_c\right) \tag{3}$$

and

$$\Delta S_T = N k_B \log\left(V_r\right). \tag{4}$$

where V_r is the relative volume ratio of the system defined as

$$V_r = \frac{V_{\min}}{V_{\max}}. \tag{5}$$

The Otto cycle was designed in 1861 to be used on four-stroke engines. The Otto cycle is similar to the Stirling cycle; the major difference comes from replacing the isothermal processes with an adiabatic process. In the case of a four-stroke engine there is also air intake and outtake processes, however we will only be considering the Otto cycle for the simple heat engine described before. With the air intake and outtake steps omitted, the Otto cycle forms a closed single directional cycle on a pressure vs volume plot shown in Fig.1(c), with η for several cycles shown in Fig.2(c). Starting at $V_{\max}$, the engine is compressed adiabatically until it reaches $V_{\min}$. The engine is then connected to the hot reservoir, allowing the body to warm up isochorically to T_h. Next the engine is expanded adiabatically until it reaches $V_{\max}$. Lastly, the engine is connected to the cold reservoir, allowing the body to cool down isochorically to T_c, ending at the beginning of the cycle. When using the Otto cycle on a simple heat engine, the Otto efficiency, η_{O}, is derived to be

$$\eta_{\mathrm{O}} = \frac{T_h\left(1 - V_r^{\frac{2}{3}}\right) + T_c\left(1 - V_r^{-\frac{2}{3}}\right)}{T_h - T_c V_r^{-\frac{2}{3}}} \tag{6}$$

Model heat engine – To model a heat engine, we created a simple environment that an agent can interact with. The environment is initialized with an engine at a volume of $V_{\max}$, and a temperature of T_c. The state of this environment is the current temperature, T, and the current volume, V, of the system. All compression and

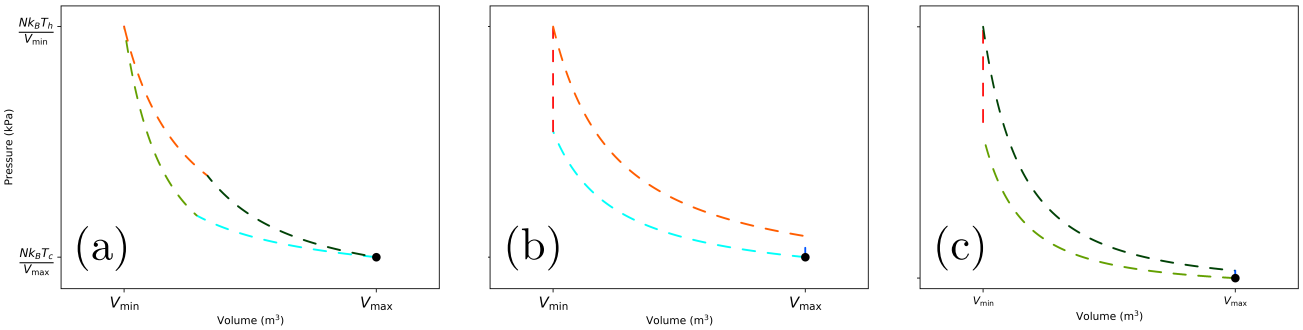

FIG. 1. The phase plot of a heat engine as it performs (a) the Carnot cycle, (b) the Stirling cycle, and (c) the Otto cycle.

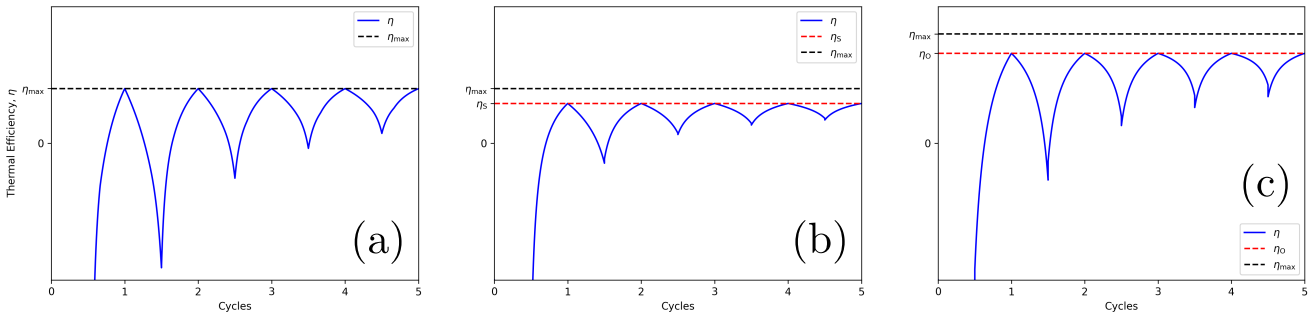

FIG. 2. η of a heat engine as it performs (a) the Carnot cycle, (b) the Stirling cycle, and (c) the Otto cycle several times each with $\eta_{\max}$ and the maximum η for each cycle for reference.

expansion actions are done using a fixed ΔV, unless otherwise stated. If an action is taken that would increase V above $V_{\max}$ or decrease V below $V_{\min}$, the state remains unchanged. After a fixed number of steps, the maximum η is used as the score of the game. To ensure the engine is usable for more than one cycle, a penalty is applied to the score of any policy that causes the engine to get stuck at a constant V. For this study, we used $V_{\min}$=2×10^{-4} m^3, $V_{\max}$=1×10^{-3} m^3, T_c = 300 K, T_h = 500 K, and ΔV values of 5×10^{-5} m^3, 1×10^{-4} m^3, and 2×10^{-4} m^3. The environment is always initialized at $V_{\max}$ and T_c. As there are no random elements in the environment itself, it is not important which V and T are used for initialization as long as it is consistent. With all actions available, the most efficient cycle possible is the Carnot cycle, therefore this first environment will be referred to as the *Carnot* environment. Using Equation 1 with these parameters, $\eta_{\max} = 0.4$.

A second heat engine environment was created, which is identical to the original one, except the adiabatic actions are unavailable. This second environment will be referred to as the *Stirling* environment as the Stirling cycle is the most efficient cycle possible with this reduced action space. Using Equation 2 with these parameters, $\eta_S = 0.291$.

A third heat engine environment was created, which is another copy of the original one, except the isothermal actions are unavailable. This environment will be referred to as the *Otto* environment as the Otto cycle is the most efficient cycle possible in this environment. T_h had to be increased to 1500 K for this environment due to the high temperatures that can be reached through adiabatic compression. Using Equation 6 with these parameters, $\eta_O = 0.658$ and using Equation 1 for comparison, $\eta_{\max} = 0.8$ in this environment.

A fourth heat engine environment was created, which includes the full action set and the agent additionally chooses which ΔV to use from: 1x10^{-4} m^3, 1x10^{-5} m^3, 1x10^{-6} m^3, 1x10^{-7} m^3, or 1x10^{-8} m^3. This heat engine environment will be referred to as the *Variable* ΔV *Carnot* environment.

Results – We first ran our GA algorithm on the *Carnot* environment using the large ΔV of 2×10^{-4} m^3. The network based policy was able to achieve a maximum η of 0.393, less than $\eta_{\max} = 0.4$. As seen in Fig.3(a), the network based policy learned a similar cycle as the one seen in Fig.1(a). Using the same network architecture, this process was repeated using the *Stirling* and *Otto* environments, yielding a maximum η of 0.291 and 0.658 respectively, the exact efficiency values as η_S and η_O.

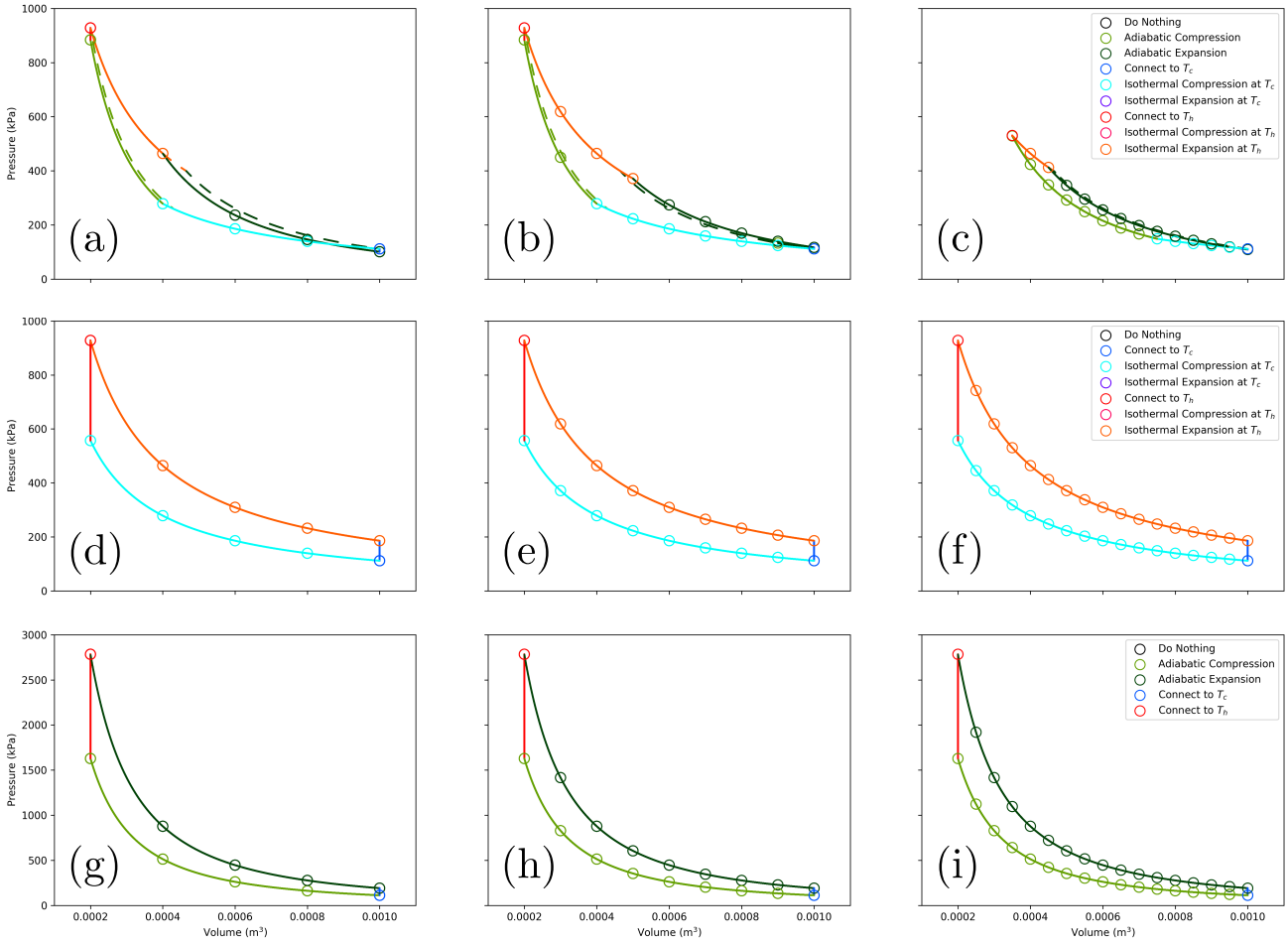

FIG. 3. Our trained network based policy agents as they act on the *Carnot* environment using ΔV values of (a) $2\text{x}10^{-3}$ m^3, (b) $1\text{x}10^{-3}$ m^3, and (c) $5\text{x}10^{-4}$ m^3 with the exact Carnot cycle for reference (dashed line), the *Stirling* environment using ΔV values of (d) $2\text{x}10^{-3}$ m^3, (e) $1\text{x}10^{-3}$ m^3, and (f) $5\text{x}10^{-4}$ m^3, and the *Otto* environment using ΔV values of (g) $2\text{x}10^{-3}$ m^3, (h) $1\text{x}10^{-3}$ m^3, (i) $5\text{x}10^{-4}$ m^3.

Unlike with the *Carnot* environment, as seen in Fig.3(d) and 3(g), the network based policy was able to reproduce the exact cycles on the *Stirling* and *Otto* environments shown in Fig.1(b) and 1(c) respectively.

Now that we have shown the network based policy performs well at a ΔV of 2×10^{-4} m^3, we reduced ΔV to more useful values of 1×10^{-4} m^3 and 5×10^{-5} m^3, then trained the network based policy GA again on each of the three environments already tested. As the network based policy was able to achieve η_S in the *Stirling* environment with a large ΔV, it should be able to achieve η_S on any ΔV which 2×10^{-4} m^3 is a integer multiple of. As seen in Fig.3(e) and 3(f), the network based policy was able to produce the exact Stirling cycle in the *Stirling* environment, with a maximum η of $\eta_S = 0.291$ as expected. Similarly, the same should be expected from the *Otto* environment. As seen in Fig.3(h) and 3(i), again the network based policy was able to produce the Otto cycle in the *Otto* environment, with a maximum η of η_O for both additional ΔV values.

Unlike the *Stirling* and *Otto* environments, it was not possible to achieve the maximum η of η_{max} in the *Carnot* environment using the large ΔV of $2\text{x}10^{-4}$ m^3. The main difference between achieving η_{max} and achieving η_S or η_O comes from the specific volumes at which certain actions need to be taken. With the Stirling and Otto cycles, actions are only ever started at V_{min} and V_{max}, where the Carnot cycle requires adiabatic actions starting at other V values, therefore it is expected that as we decrease ΔV, the maximum η our agent can achieve in the *Carnot* environment will increase. Using a ΔV of $1\text{x}10^{-4}$ m^3, the network based policy was trained on the *Carnot* environment again, yielding a maximum η of 0.398, higher than the maximum η found when using a ΔV of $2\text{x}10^{-4}$ m^3. As seen in Fig.3(b), the cycle produced by our agent using this smaller ΔV more closely resembles the actual

Carnot cycle, however it is still not the exact cycle, therefore ΔV was decreased again to 5x10^{-5} m^3. With this even further decreased ΔV on the *Carnot* environment, the network based policy was able to achieve a maximum η of 0.3993, even closer to $\eta_{\max}$ than with the previous ΔV. As seen in Fig.3(c), unlike every other case seen so far, the optimal cycle with this ΔV does not use the full available set of volumes. This shows that, unlike the Stirling and Otto cycles, the volume of our system is not important for η. What is important for maximizing η in our *Carnot* environment is being able to go from T_c to T_h without isochoric actions after isothermally compressing the system by some amount, and being able to go from T_h to T_c without isochoric actions after isothermally expanding the system by some amount. For this reason, to achieve a maximum η of $\eta_{\max}$, ΔV must be small enough that the system can reach the exact V values required for the adiabatic actions to be started.

To fit the data produced by the agent acting on the heat engine environment we used a function flexible enough that it can be used to fit both ideal and van der Waals gases for, isothermal, adiabatic, and irreversible compression and expansion as well as isochoric heating and cooling. The function $P(V,T)$ is

$$P(V,T) = \frac{nRCT^{x_1}(1-k)^{f(V)}}{V^{x_2} - nb} - \frac{an^2}{V^2}, \tag{7}$$

where n is the number of moles of the gas, R is the gas constant, C is a general constant, x_1 is the Boolean exponent which determines if T is used in the equation, x_2 is the volume exponent which is either 1 or γ for the gas, a is a constant specific to the gas, and b is the volume per mole that is occupied by the molecules. To optimize this equation for a specific segment of the a thermodynamic cycle, x_1, x_2, a, and b are discretized and iterated over while C and k are fit using a least squares method for each x_1, x_2, a, and b group.